EVOLUTION OF THE SOFTWARE COMMUNICATION ARCHITECTURE STANDARD

Kevin Richardson, Chalena Jimenez, Donald R. Stephens
Joint Program Executive Office (JPEO), Joint Tactical Radio Systems
San Diego, CA

ABSTRACT

Three primary objectives of the Joint Program Executive Officer (JPEO) Joint Tactical Radio System (JTRS) are: a) reduce time to field capability, b) improve interoperability between services, and c) decrease radio production costs. The Software Communications Architecture (SCA) specification is the architectural framework for all JTRS software artifacts that was created to maximize software application portability, reusability, and scalability while providing the flexibility to address domain specific requirements.

SCA version 1.0 was published in 2000 and the last major release (version 2.2.2) was published in 2006. Since release of version 2.2.2, only minor enhancements have been made to the specification, which have been targeted towards addressing items of immediate concern to the development of the Increment 1 JTRS products. Over the course of the last two years technologies have evolved and there have been many SCA related lessons learned through the development of JTRS products such as Ground Mobile Radio (GMR). JPEO JTRS asserts that the SCA will continue to evolve so that JTRS products meet the current and emerging needs of the next generation warfighter.

To address the requirements of the JTRS stakeholders, JPEO JTRS has initiated the development of a new SCA release. The overriding philosophy behind this revision is to position the SCA as a specification that is comprehensive yet flexible enough to provide a technical foundation for multiple generations of JTRS and industry products. To accomplish this flexibility, the proposed SCA enhancements will migrate the specification towards a technology independent representation and away from the current dependence on Common Object Request Broker Architecture (CORBA) and eXtensible Markup Language (XML) Document Type Descriptor (DTD) files. A second feature will introduce optional elements within the specification so that compliant products may be developed which better map to the functional and resource requirements of a wide array of target platforms.

Origin of the SCA

The JTRS Enterprise Business Model is based on competing vendors for production of all Joint Tactical Radio (JTR) sets and interoperability. These attributes initiated the decision for waveform portability between radio sets. To achieve this, the JTRS Program Office (JPO) established the SCA.

The SCA was initially developed in steps [1]; in the first step three consortia provided studies on the architecture. The studies were similar, and in the next step the technical evaluation team incorporated 'golden nuggets' from the studies into an award to write the SCA standard, implement the SCA and provide a proof of concept effort. The final steps provided additional SCA validation and proof of concept efforts.

The primary architecture development areas were a) modular functional breakout, b) well-defined functional entities, c) rule sets for functional entities and implementation guidance, d) open standards, e) domain independence, f) system control, g) software development and distribution, and h) interface descriptions. With this guidance, the consortium began development of the SCA and delivered version 0.1 in December 1999. The early SCA versions contained the SCA “Core Framework” along with its appendices, the Application Program Interface (API) Supplement which started the process of defining common APIs that could be shared across JTRS products and the Security Supplement which introduced a collection of requirements which could be leveraged to develop DoD communications products. The original SCA software structure is shown in Figure 1 Original SCA Software Structure.

Many refinements and versions followed, with the SCA 2.2 specified for the first radio product release.

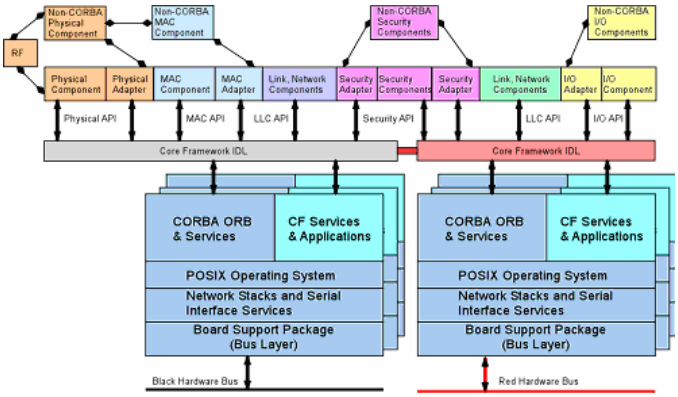

Figure 1 Original SCA Software Structure

SCA 2.2.2 deprecated the API Supplement and moved the API standardization and definition processes to a different development initiative within the JPEO. In addition, this SCA version did not incorporate the Specialized Hardware Supplement that was defined in an earlier SCA release. Many of the Specialized Hardware Supplement capabilities were captured within the JTRS Interface Control Working Group (ICWG) Approved JTRS Standard APIs [4]. The Security Supplement was also deprecated in favor of existing specifications. This series of changes was made so that the SCA could maintain its position as a specification that could be leveraged by both government and industry. When SCA elements were removed great care was take to ensure that JTRS Products could still extend the fundamental architecture to provide similar functionality.

In December 2006, an extension was added to introduce additional capabilities requested by the radio programs. The SCA 2.2.2 extension provided a mechanism to optimize application deployment on multi–channel environments. It also supported the definition and deployment of non-SCA services (i.e. services other than Log, FileSystem, Event, and Naming). The extension introduced new requirements for the SCA 2.2.2 Core Framework but only for those radio platforms that wanted to take advantage of this additional capability. The concept utilized for this extension was similar to the design pattern used for API extensions [3].

In December 2007, the Application Environment Profile (AEP) amendment updated the Portable Operating System Interface for Unix (POSIX) profile to incorporate more lessons learned from the radio programs and current computing standards.

Current SCA

Although the word architecture is part of its name, the SCA is not a complete architecture – it is considered an architecture layer or framework for a software defined radio. The SCA consists of specific base interfaces, operating environment specifications, design patterns, and specific radio utilities as depicted in Figure 2 The Many Aspects of the SCA.

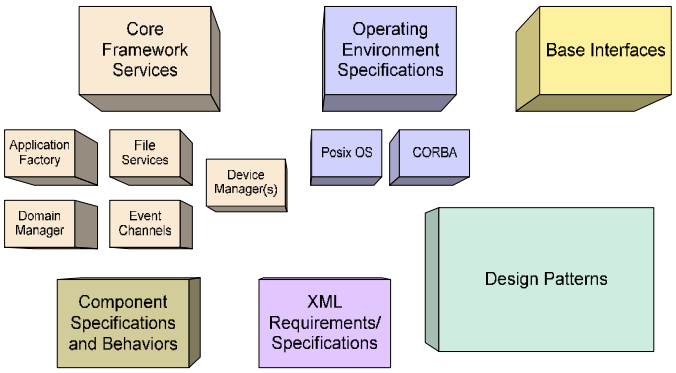

Figure 2 The Many Aspects of the SCA

The SCA provides one component of the end architecture that is developed for a software defined radio. The JTRS program has supplemented the SCA with a collection of APIs that can be used to extend the SCA component based framework. The collection of SCA and JTRS APIs provides a more fully refined definition of a complete host environment to execute portable waveforms. The fact that the SCA is not a full architecture is a key attribute of the SCA and not a weakness. This permits the SCA to be incorporated into the architectures of radios ranging from a small battery-operated sensor to a multi-channel, reconfigurable airborne radio platform as illustrated in Figure 3 JTRS Radios Enabled by the SCA.

Figure 3 JTRS Radios Enabled by the SCA

Operating Environment

The SCA is intended to be applicable for a multiplicity of communications devices; supporting the requirements of a wide range of military, civilian and commercial platforms. The JTRS program desires one waveform to be portable across all the radios illustrated in Figure 3 JTRS Radios Enabled by the SCA.

To achieve this, the waveform developer must be limited to a subset of operating system and programming language features. Similarly, the radio set must be able to support these exact functions and capabilities with its processor and battery.

The AEP of the latest SCA release tailors more than 500 specific real-time operating system functions that must be supported by an SCA-compliant product as illustrated in Figure 4 Portion of POSIX Functionality Specification. The AEP is a hybrid POSIX real time profiles PS 52&53, however it is supported by multiple real-time operating systems vendors. The SCA specification allocates the Real-Time Operating System (RTOS) requirements to the operating environment so a JTRS product developer has the option of supporting the AEP requirements via commercially available and proprietary products. The AEP requirements are only applicable to General Purpose Processing (GPP) processing elements as POSIX compliant implementations are not as ubiquitous for Digital Signal Processing (DSP) processors. JTRS has published guidelines for non-GPP processing elements, but not specified run time operating environments.

Function	AEP	Function	AEP
fesetenv()	NRQ	isdigit()	MAN
abs()	MAN	isgraph()	MAN
asctime()	MAN	islower()	MAN
asctime_r()	MAN	isprint()	MAN
atof()	MAN	ispunct()	MAN
atoi()	MAN	isspace()	MAN
atol()	MAN	isupper()	MAN
atoll()	NRQ	isxdigit()	MAN
bsearch()	MAN	labs()	MAN
calloc()	MAN	ldiv()	NRQ
ctime()	MAN	llabs()	NRQ
ctime_r()	MAN	lldiv()	NRQ

Figure 4 Portion of POSIX Functionality Specification

The CORBA has been specified from the very early versions of the SCA. CORBA provides support for two important SCA attributes; distributed processing and a component model. The Object Management Group's (OMG) Interface Definition Language (IDL) is used for the base SCA interfaces as well as the separately defined JTRS APIs. Software tools exist to generate software code skeletons from the IDL definitions that can be populated with behavior logic.

Because many software defined radios have multiple processors, the SCA addresses interprocessor communication. Without such a specification, it becomes impossible to achieve waveform portability across the product line depicted in Figure 3 JTRS Radios Enabled by the SCA.

CORBA provides a set of transport protocols, an intermediate data representation and a mechanism to disassemble and reconstruct method invocations in support of this distributed procession requirement.

Three additional services are provided by CORBA, the naming service, event service and log service. The existence of these services relieves the SCA from having to define these components. The OMG naming service performs the function of a telephone book for software and hardware adapters within the radio set. A software component registers with the naming service and then other components can query the naming service to obtain the communication address of that registered component. The OMG log service standardizes all logging within the radio. Components can log events and messages following the open standard format and the radio set infrastructure transparently assumes responsibility for the lifecycle of the component. The OMG event services provides a decoupled communication mechanism for software components to signal either the radio set or other software components using a publish-and-subscribe open architecture interface.

Component Model and Dynamic Instantiation

In addition to the component model provided in the CORBA specification, the SCA provides a dynamic instantiation and connection of software radio components as illustrated in Figure 5 Dynamic Instantiation and Connection. The SCA does not assume a static configuration of components. When a radio powers on, the software components in Figure 5 Dynamic Instantiation and Connection are launched by the radio's boot loader, but there is no connectivity between these components and they have no *a priori* knowledge of how to communicate with each other.

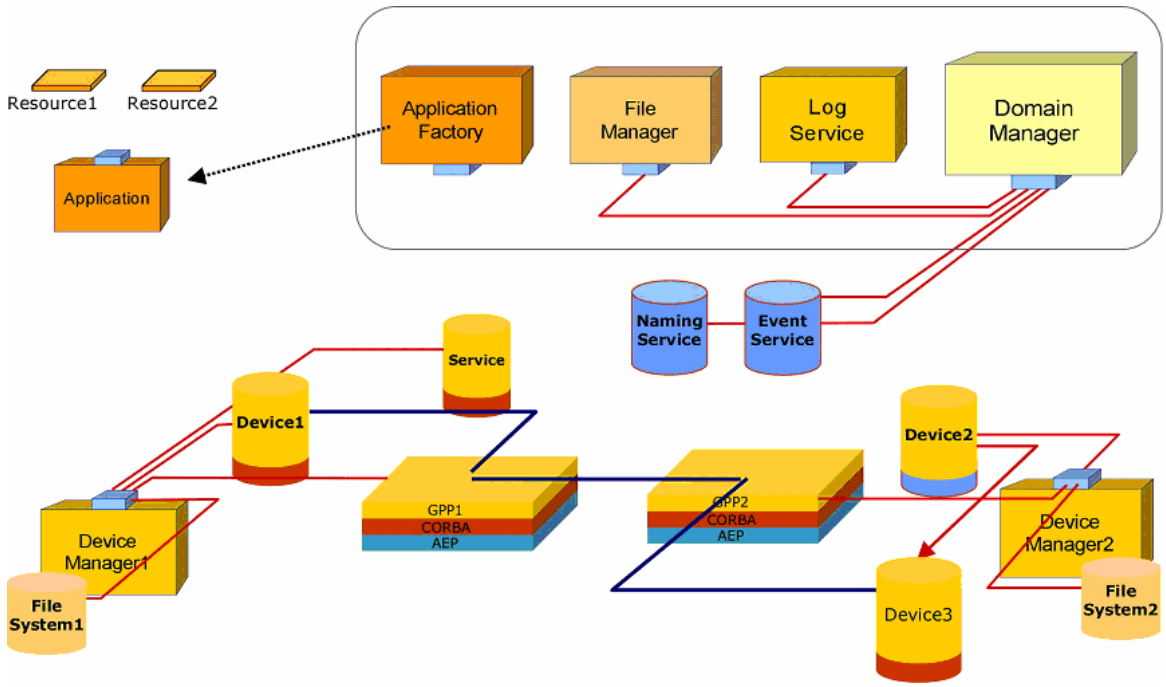

Figure 5 Dynamic Instantiation and Connection

The individual components expose CORBA interfaces on ports, and the Domain Management interfaces (Domain Manager, Application Factory and Device Manager) read XML files that provide the names of the port connections for all of the software components. Similar to a hardware schematic diagram, the XML files permit the Domain Management interfaces to initiate connections between all of the radio components. When a waveform is launched, the Application Factory will perform a similar connection activity for the waveform components, shown as resources in Figure 5 Dynamic Instantiation and Connection. It will also initiate connections between the radio set components and the waveform components as specified in the waveform's XML files.

The SCA also specifies a set of base interfaces illustrated in Figure 6 Base Interfaces Established by the SCA. Because every software component and corresponding hardware interface is required to implement a *CF::Resource* interface, the SCA establishes a common management and control mechanism for the radio set.

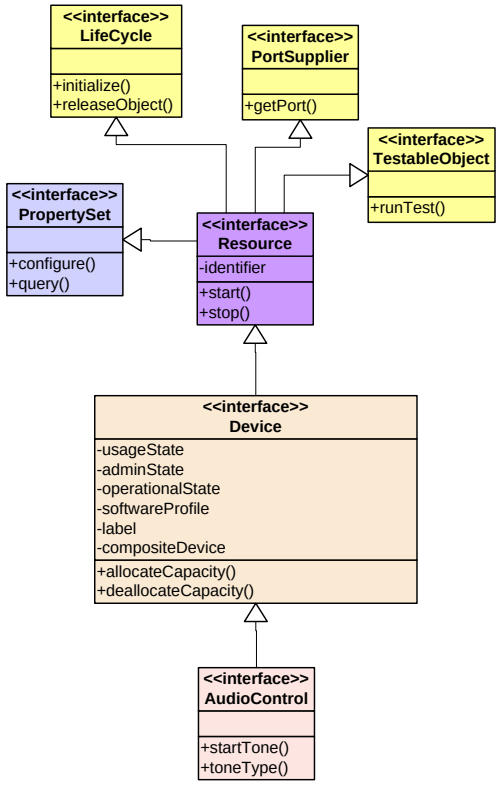

Figure 6 Base Interfaces Established by the SCA

Success of the SCA

JTRS is experiencing great success in porting new networking waveforms such as Soldier Radio Waveform

(SRW) and Wideband Networking Waveform (WNW) to new radios. The recent Airborne, Maritime Fixed (AMF) radio was contracted without any waveform development. The contractor will download JTRS software from the JTRS Information Repository and refactor it slightly for that set's unique mission requirements and configurations; instead of completely re-developing the waveform.

Goals for SCA Evolution

The JTRS ICWG is currently developing proposals to evolve the SCA from the current SCA 2.2.2 version. The ICWG is composed of technical and programmatic representatives that span the entire spectrum of JTRS Programs, development contractors and key stakeholders. Figure 7 Different SCA Requirements From Different Form Factors illustrates some of the opposing requirements of the SCA for different radio form factors and missions. The battery-powered sets need to conserve battery life; hence any reconfigurability that costs memory or CPU cycles is unnecessary and counter to mission requirements. Radio sets that have a prime power source can utilize the flexibility and reconfigurability of the current SCA. An expanded feature set is desired, especially for the radio sets that require extensive platform integration such as airborne radios integrated into an airframe's aviation suite.

SCA Evolution Process Overview

The ICWG has had a key role in defining the scope of the SCA enhancements, but the issues that will be addressed are a byproduct of lessons learned from SCA compliant products developed by JTRS Programs, Industry and Academia. Throughout the history of the JTRS Program the Program Office has made it a priority to maintain a close relationship not only with its product developers but also with industry and academia through forums such as OMG and the Software Defined Radio Forum (SDRF). In addition to maintaining contact it has also been a priority for the program to capture and record data regarding areas the SCA could be enhanced or expanded.

One of the first activities in the evolution process was to consolidate the outstanding enhancement requests and then augment those with suggestions from the ICWG membership. After that point an SCA Program Team was established whose responsibility is to manage and execute the development of revised SCA content. The SCA Program Team is comprised of individuals who are representative of the entire ICWG membership. Those individuals are required to be empowered to represent the interests of their constituents, identify appropriate resources to develop individual enhancements and critique and refine the issue resolutions proposed by other SCA Program Team members.

The SCA Program Team prioritized the set of approximately 60 enhancement proposals. The items were prioritized according to their perceived benefit to the end users of the SCA. The prioritization was not based on the extent of the modification, so highly rated changes could simply require a few wording modifications or entail changes that affect major philosophical positions.

The result of the prioritization activity was the selection of roughly 20 enhancement proposals that were identified as “must have” topics to be investigated as candidate additions for the next SCA version. The remainder of the paper explores some of the enhancement proposals that will be considered during the SCA evolution process.

Candidate SCA Enhancements

In addition to some of the attributes identified in Figure 7 Different SCA Requirements From Different Form Factors, an objective of the SCA evolution will be to maintain backward compatibility for products developed for the existing SCA specifications since the JTRS Information Repository contains 4.5 million lines of source code compliant to SCA 2.2.2.

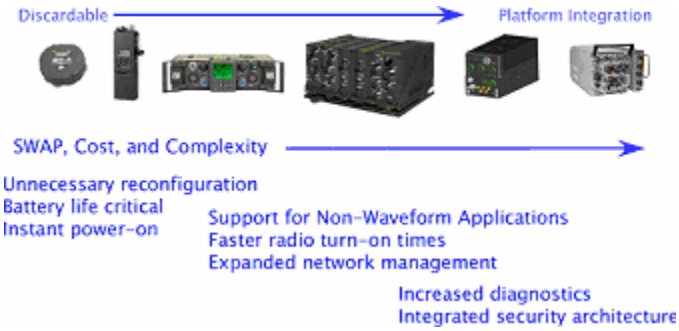

Figure 7 Different SCA Requirements From Different Form Factors

To support the opposing requirements of Figure 7, the revised SCA will be more configurable to allow for a wider range of products to be viewed as compliant. The SCA will identify more compliance points that will allow for multiple views of what constitutes a conformant implementation. For example an existing SCA Device is a CF::Resource that fully supports a state model and has the ability to manage allocation

capacities. A plausible system architecture could include a Device that does not need to have the ability to manage capacities. Perhaps the device in question is used as a communication relay that is used to transition data messages between communications endpoints. For this scenario, the system designer may have determined that the manner in which the target system aggregates data and sends it across the relay is sequential in nature and consequently it would be impossible for the capacity of the device to be exhausted. An SCA 2.2.2 compliant device would have to provide a full implementation of the Core Framework capacity management operations, allocateCapacity() and deallocateCapacity() which have several semantic requirements for expected behavior and exception cases. The revised specification will introduce capacity management as a configurable capability. If a device needs to be a capacity manager then it can incorporate the capability and be subject to the existing Core Framework requirements, however in cases such as the one described above, the framework will provide a mechanism to omit the capability and thus the device will not be subject to those requirements. By following this pattern, existing implementations can maintain their compliance with little impact, but the incorporation of the additional flexibility will allow the framework to be applied in a manner that will be better aligned with the target system requirements.

Requirements Analysis and Allocation

SCA 2.2.2 combines requirements belonging to radio set providers (i.e. Operating Environments), and Waveform Application developers. The specification attempts to categorize the requirements through terminology such as Base Application and Base Device interfaces; however the distinctions may be lost on all but the savviest JTRS developers. The revised specification will be refactored so the responsible owner of any particular requirements is clear and precise. The allocation can be coupled with the compliance declarations so that even a casual reader of the specification should be able to easily identify which portions of the document are applicable to their desired products. This will aid certification testing and also improve the productivity of developers since requirements ownership will be well defined.

Requirements will be examined to determine whether they are critical or 'nice-to-have'. Those belonging to the latter category will be dropped. The emphasis of this activity is specifically targeted towards optimizing the system so that it better aligns with the constraints levied by real time systems.

CORBA Neutral Migration

Most Waveform Application developers are comfortable with the AEP specifications, particularly since they have been recently updated to the latest IEEE standards. CORBA, however, may be costly in terms of memory and CPU cycles for battery-powered sets. There are other issues with CORBA such as its non-determinism and its lack of compatibility with certain secure computing principles.

Unfortunately, there is not a one-for-one replacement readily available. Some technologies such as remote procedure call (RPC) accommodate distributed processing, but lack a component model. There are more expressive technologies such as Simple Object Access Protocol (SOAP) but they are not compatible with real time systems and in some cases may require development of a number of the services that are included within most Object Request Broker (ORB) implementations.

A great deal of the value of the SCA comes from the abstractions that are embedded within the framework and the manner in which they can utilize commercially available open technologies to facilitate the development of portable Software Defined Radio (SDR) systems. Hence, one of the major benefits gained from SCA development is the specification of the common model. Once the model is captured then the specification can be tailored at many levels to emphasize whatever is important to the tailoring organization. A program could give its developers the latitude to use whatever technologies best suited their product as long as they preserved the SCA model, the program could dictate the complete set of platform specific technologies to be leveraged within their program, or there could be an intermediate solution anywhere between those two endpoints. This approach accommodates existing SCA compliant implementations as well that those which may choose to migrate to CORBA/e, the nominal successor to minimum CORBA.

The IDL representation will be preserved as the technology specific standard way to express the specification’s interfaces; however there is an expectation that as the revised specification matures, a collection of mappings will be developed and standardized to transform that neutral representation into additional technology specific representations.

Implementation Guidance

Many of the APIs in the SCA specify exception processing. However, the SCA does not mandate native C++ exception handling. Although convenient because it is a feature of the language, many embedded programmers shun it because of the memory and CPU costs. Similarly run-time type information (RTTI) and dynamic casting are features of C++ that are very costly in resource-constrained, embedded platforms.

Waveform developers need guidance on these programming language features. Without guidance, a Waveform Application developer could assume a resource-rich platform and produce a Waveform Application that would be unable to execute on the less-capable radio sets as depicted on the left-hand side of Figure 7 Different SCA Requirements From Different Form Factors.

Similar to the technology specific mappings that will be associated with the SCA, many of these guidelines will be specific to a particular technology representation. The JTRS program intends to develop this type of guidance, but it will beyond the scope of the SCA.

Optional Framework Controls

The smaller, resource-constrained radio platforms may have less need for the existing capabilities of the SCA Framework Control Interfaces, particularly those that provide dynamic instantiation capabilities. Such aspects will be addressed during the detailed analyses that will be executed as part of the evolution process. The two most likely outcomes for this evolution are for the capabilities to be provided in the SCA via an extension mechanism similar to that provided for the APIs [4], permitting radio set developers to select the elements most applicable to their mission. The second option would be for the functionality to be provided via a configurable capability expressed within the framework.

Evolving The Domain Profile

The SCA Domain Profile is currently described by a set of XML DTD files. There is current investigation into whether we should move from the use of DTDs to XML Schema (XSD) files to provide more flexibility to the user. XML Schema files provide an alternate but equivalent mechanism to validate the identity, capabilities, properties, and inter-dependencies of domain components. In addition, the XML Schema provides added features that may be of use when evolving the SCA. There are minimal changes that will need to occur to validate existing component files with XSD verses DTD files. Further work is underway to validate the XML to XSD mapping and to determine if there are any impacts with using the files.

The SCA Program Team will also investigate whether or not platform independence will be extended to the descriptor files as well. If that ends up as the case then descriptor formats beyond the XML based alternatives will be considered for inclusion and the XSD might be established as the standard from which future technology specific mappings are spawned.

Conclusion

The SCA has gone through a number of different stages through its existence. The SCA has proven itself as a flexible architectural framework that has been used in the development and operational deployment of a variety of SDR products. The JTRS program is undertaking an effort to extend the lifespan of the SCA far into the next generation of configurable, programmable SDR communication devices. In addition to the topics that were addressed in this paper other topics such as the role of Services within and SCA compliant implementation and enhancements to the application deployment processes will also be addressed as part of the evolution activities. In order to achieve that objective JPEO JTRS is taking a series of actions targeted towards emphasizing the framework's common and reusable strengths. Qualities such as increased configurability and CORBA neutrality are critical towards increasing the flexibility of the specification such that it can accommodate the breath of functional and non-functional requirements levied by future communications devices.

References

[1] J. Place, D. Kerr and D. Schaefer Joint tactical radio system, MILCOM 2000 - IEEE Military Communications Conference, no. 1, October 2000, pp. 209 – 213

[2] Donald R. Stephens, Brian Salisbury, Kevin Richardson, "JTRS infrastructure architecture and standards", MILCOM 2006 - IEEE Military Communications Conference, no. 1, October 2006 pp. 3481-3485.

[3] Donald R. Stephens, Cinly Magsombol, Chalena Jimenez, "Design patterns of the JTRS infrastructure", MILCOM 2007 - IEEE Military

Communications Conference, no. 1, October 2007 pp. 835-839.

[4] Cinly Magsombol, Chalena Jimenez, Donald R. Stephens, "Joint tactical radio system—Application programming interfaces", MILCOM 2007 - IEEE Military Communications Conference, no. 1, October 2007 pp. 855-861.